# THE SHELL GAMES

## EDUCATION SERIES


apple® computer inc.

NOTICE

Apple Computer Inc. reserves the right to make improvements in the product described in this manual at any time and without notice.

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

APPLE COMPUTER INC. MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL OR WITH RESPECT TO THE SOFTWARE DESCRIBED IN THIS MANUAL, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. APPLE COMPUTER INC. SOFTWARE IS SOLD OR LICENSED "AS IS". THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE IS WITH THE BUYER. SHOULD THE PROGRAMS PROVE DEFECTIVE FOLLOWING THEIR PURCHASE, THE BUYER (AND NOT APPLE COMPUTER INC., ITS DISTRIBUTOR, OR ITS RETAILER) ASSUMES THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION AND ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES. IN NO EVENT WILL APPLE COMPUTER INC. BE LIABLE FOR DIRECT, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE SOFTWARE, EVEN IF APPLE COMPUTER INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.



© 1979 by APPLE COMPUTER INC.
10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010

All rights reserved.
APPLE product # 030-0066-00

# THE SHELL GAMES
## EDUCATION SERIES

by Bruce Tognazzini

# TABLE OF CONTENTS


1 Introduction

2 Getting Started

3 The Match Machine

4 Professor True

5 Mr. Multiple

6 Creating Your Own Quizzes
6 Education & Design Philosophy
7 Before You Begin
8 Special Note on the Disk II

9 Inside The Match Machine
10 Anatomy of Shell Game Problems
11 Specific to The Match Machine
13 Entering a New Match Problem

16 Entering Problems
16 The Problem Entry Editor
16 Entering a New Problem
20 Editing an Existing Problem
22 Displaying a Problem
23 Searching For a Problem
24 Verifying the Data Table
25 Replacing an Old Problem
26 Inserting & Deleting Problems
27 Leaving the Editor
28 An Important Fringe Benefit
28 Fitting Your Program Needs

30 Inside Professor True
31 Data Structures & Scary Stuff
34 Modifying Professor True

37 Inside Mr. Multiple
39 General Comments
39 Modifying Mr. Multiple
40 In Conclusion

41 Appendix A:
Formats & Special Symbols
41 For All Shell Games
42 The Match Machine
43 Professor True
45 Mr. Multiple

47 Appendix B:
Bibliography

# INTRODUCTION

Welcome to The Shell Games, a group of quiz game programs that, while fun to play, are actually educational tools. The Shell Games are so called because the games themselves are "shells" that you can put your own quizzes into. The questions, answers, and matches the Games currently contain may be removed and new ones inserted quickly and easily. The function of the programs is to take care of giving instructions (and encouragement) to the user, formatting text, and keeping track of details that can take up so much of your time in preparing computer quizzes. In addition, the programs share an editor designed to make problem entry fast and accurate.

# GETTING STARTED

Before you begin, you should go through chapters one through four in the Do's and Don'ts of DOS manual that you received with your disk drive. These chapters give basic information on using the Disk II and the most recent version of DOS.

Diskettes, though protected by tough cardboard sleeves, are delicate creatures. They can be accidentally destroyed, and the information on them can be written over if they are not write-protected. You should always make back up copies to use and write-protect your originals (see your DOS manual), then store the originals in a safe place. In fact, it is a good idea to keep two backup copies in case something really drastic happens and one or both of your backups are ruined.

The backing up procedure is not difficult. There are two ways to do it. If you have one disk drive you will have to copy the programs one at a time. If you have two or more disk drives, you can use the COPY program on your System Master diskette to copy the entire diskette at once.

To copy The Shell Games diskette using one disk drive, first boot DOS and then initialize a blank diskette using the procedure given in your DOS manual. When you've done this, insert the Shell Games diskette and use CATALOG to see what programs are on it. There should be six programs listed in the catalog. They are

HELLO
INDEX
THE ANIMATED APPLE
THE MATCH MACHINE
PROFESSOR TRUE
MR. MULTIPLE

LOAD the first program in the catalog (called HELLO), insert your freshly initialized blank diskette, and SAVE the program, using the same name. Repeat this procedure for all six programs, using CATALOG to check that all the programs have been transferred succesfully.

If you have more than one disk drive, simply RUN the program called COPY from the SYSTEM MASTER diskette that came with your disk drive. Instructions for using this program are in your Do's and Don'ts of DOS manual.

When you have successfully backed up The Shell Games diskette, put the original copy in a safe place, and then put the backup copy into your disk drive and boot it.

# THE MATCH MACHINE

Program Language: Integer BASIC
Minimun Memory Requirements: 32K

Whir, click, spleedle, bleen,
Here comes the Magnificent Match Machine!
Chomping up matches and stirring them, then,
Watching you put them together again!

Pick out a subject, whatever seems best,
One-half of the match will appear while the rest,
Gets shuffled and shaken and stirred round and round
In the match machine mixer with great greeble sound.

With blip, bloop, and blobble, the matches pour out,
They fill up the screen. With excitement you shout,
"Magnificent Match Machine, match me a match!"
And spying an answer, the keyboard you snatch.

The number you type of the choice you would mate,
Then you press on a letter and sit back and wait.
While Apple decides on the course of your fate.
"Oh, it has to be wrong!" But now it's too late.

The screen has erupted in red, yellow, blue!
The match machine says that the winner is You!
A match that's correct on your very first try!
You're certainly clever, more clever than I!

And so you continue with blip, bloob, and bleen,
As you run the Magnificent Match Machine.
Until at long last all the quizzes are done,
Until at long last all the quizzes you've won.

# PROFESSOR TRUE

Program Language: Integer BASIC
Minimum Memory Requirements: 48K

Gentle Reader,

There is a Gentleman up with whom I cannot put, a gentleman who has had the unmitigated gall to recklessly plagiarize my work and claim it for his own. A man who was given 24 hours to get out of town after sending the Dean of our mutual Alma Mater, O.S.B.U.T.S.G.E* , a most scurrilous epistle scrawled on the side of a mule. I am speaking of that minor-league villian, Mr. Multiple.

It was he who snuck into my computer room a few weeks ago while I was in the Laboratory testing a solution of the Klarn paradox. I emerged from the cellar just in time to see him scurrying away, a diskette dangling perilously from beneath his cloak. The very next day he held the historic press conference on the steps of the Farnsfarfle project (see: The Infinite Number of Monkeys, Contributed Programs Volumes 3-5), announcing his creation of the Mr. Multiple program.

I urge you to boycott Mr. Multiple. Not only are his questions open to question, they are offensively silly when compared to my own brilliant work. (Besides, I award a lot more points in my quiz!) Professor True openly displays my obvious genius for clarity.

I would like to acknowledge the work of the brilliant research team of Hank Suchorski and J. Alfred Glitch, without whom this program would have been possible.

--Professor True

*O.S.B.U.T.S.G.E. is, of course, that bullwork of New England scholastiscm, Old Sow Belly University and Truck Stop - Good Eats.

# MR. MULTIPLE

Program Language: Integer BASIC
Minimum Memory Requirement: 48K

Dear Apple Person,

To think that a man of my reputation would have to stand up to such ridiculous charges as those leveled by my old college chum, "Professor" True! I deny the allegations and I defy the allegator!

The fact is that the Klarn solution the Professor was testing is about 190 proof. It was developed back in the 30's by Mortimer Starzynski (see: The Great American Probablity Machine, Contributed Programs Volumes 3-5), said to have passed away a few years ago, but actually living in California under the assumed name of Hank Suchorski. True was in no condition to assess the situation correctly; what actually occurred was that I dropped a fresh copy of Mr. Multiple after helping him stumble up from the cellar. He apparently found it and changed it to reflect his own dull tastes. His version is to be avoided at all costs!

Mr. Multiple

# CREATING YOUR OWN QUIZZES

The Apple makes learning fun; it should make teaching just as interesting. Designing a well thought-out quiz is difficult and creative work. We want to make it easy for you to enter your quizzes into the Apple, as well as edit or add to them once they are there.

# EDUCATION & DESIGN PHILOSOPHY

Not to be read by kids! (If you do, a carrot will grow out of your left ear! Please skip down to "BEFORE YOU BEGIN".)

Before delving into the secret world of the quiz editor (called the Problem Entry Editor), here are a few words about the educational philosophy of The Shell Games. You will notice a maximum amount of positive reinforcement and an absolute minimum of negative sanction. As the quizzes were designed primarily to teach, not test, there is no penalty for missing a question. But the user is encouraged to take the missed questions over again until they are answered correctly, thereby receiving the color and sound "reward".

The "backward" problem format of Professor True and Mr. Multiple, with the explanation following the question, has proven to be very effective: on new information, the user will often miss the first time, but an intriguing question will raise interest in the answer which follows. The student's comprehension and retention the second time around has proven to be very high. And the same users a week or two later have continued to show very high retention of the information. This has proven true even for children in the very early grades.

If you are a teacher, you will soon discover two methods of entering new quizzes into the games. One is by using the computer, as will be described below. The other is by having the students do the entry! This has several advantages, not the least of which is a savings in time for you. Beyond that, however, you will be providing your students with a chance to develop their computer literacy by doing useful programming to which they will be able to point with pride. And they will not only learn more about computers, but will automatically learn the contents of the quiz. Consider having your students not only enter but create the contents of quizzes to be used by next year's class. This will be an exercise requiring full knowledge of the subject about which they will be constructing questions -- they will eagerly learn the subject matter because it is to their immediate advantage to do so. Try it and see. It works!

The Shell Games are to be used by very young students as well as adults so it is important that they be as easy to use as possible. For this reason the "data" for The Shell Games (your quiz problems) become part of the program itself, so a user need only do a standard program LOAD. When you change problems in a Shell Game, you need only SAVE the program again to have your new version.

# BEFORE YOU BEGIN

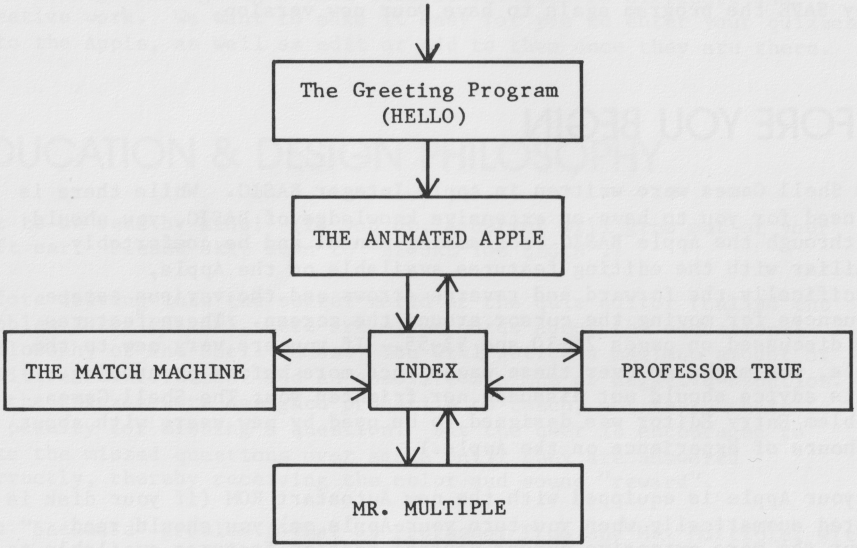
The Shell Games were written in Apple Integer BASIC. While there is no need for you to have an extensive knowledge of BASIC, you should go through the Apple BASIC Programming Manual and be comfortably familiar with the editing features available on the Apple, specifically the forward and reverse arrows and the various escape sequences for moving the cursor around the screen. These features are discussed on pages 28-30 and 53-55. If you are very new to the Apple, you might go over these pages once more before proceeding. (This advice should not dissuade nor frighten you: The Shell Games Problem Entry Editor was designed to be used by new users with about 10 hours of experience on the Apple.)

If your Apple is equipped with the new Autostart ROM (if your disk is booted auomatically when you turn your Apple on) you should read about the more extensive cursor control editing features available to you. This information is available in both The Applesoft Tutorial and the Autostart ROM manual.

Because you will find the same Problem Entry Editor in each program in The Shell Game series, we will not repeat the full explanation of the Problem Entry Editor for each game. Instead, the following documentation has been designed to carry you gently through from the simplest game (The Match Machine) to the most advanced, step-by-step, pointing out in the later games only the added features and powers. Therefore, please begin with us at the beginning and try each step on the computer as we go along.

# SPECIAL NOTE ON THE DISK II

The programs on the diskette are arranged in the following way:




Upon booting the diskette, the Hello program immediately RUNS The Animated Apple which in turn RUNS Index, the table of contents. From Index, the user may go to any of the four programs, returning to Index at their completion, or when desired by pressing ESC or following other directions out of the program. If at any time you stop a program in progress (by using a CTRL-C ), you may return to the main menu by typing:

RUN INDEX

This will eliminate from memory the program you stopped with the CTRL-C and RUN the Index program. (As you will soon see, it is not always desirable to eliminate the current Shell Game from memory: if you modify a Shell Game and wish to keep the changes you've made, you must SAVE the game before RUNning Index.)

# INSIDE THE MATCH MACHINE

Please LOAD The Match Machine from diskette by selecting

2 THE MATCH MACHINE

from the main menu. Then, as soon as the title page appears, type CTRL-C (while depressing the CTRL key, press C , then release both) thus stopping the program.

Now type:

LIST 9010, 9014

Listing these lines will display a sample of the program that produces each set of matches. These are the portions of the programs with which you will be working when you create games of your own, so be sure you understand what we are describing.

If you have the original quiz, you will see:

```
9010 D1$="ANIMALS AND THEIR YOUNG,ANI
     MAL&ITS YOUNG,BEAR&CUB,BIRD&NEST
     LING,CAT&KITTEN,COW&CALF,DOG&PUP
     PY,FISH&FRY"
9011 D2$="FROG&TADPOLE,GOAT&KID,KANGA
     ROO&JOEY,SHEEP&LAMB,BUNNY,KIT,EL
     VER": RETURN
9012 REM
9013 REM
9014 REM
```

It doesn't matter if you find different lines; just refer to the lines on the previous page during the explanation.

Once you understand the above 5 lines and why they look as they do, you will have grasped 95% of what you need to know to write your own quizzes. So those of you who are new to programming -- DON'T PANIC!

# ANATOMY OF SHELL GAME PROBLEMS

The general anatomy of the problems is shared by all Shell Games. There are eight principles to that anatomy:

1. The first problem in all Shell Games can always be found beginning on line 9010. (There are more Shell Games being written.)

2. All Shell Game problems occupy exactly five consecutive lines: 9010 to 9014 for the first problem, 9015 to 9019 for the second problem, 9020 to 9024 for the third problem, etc.

And now, about the strings. [Oh, no, Henry, he's going to talk about strings! Fetch me my powders!] A string, as you will recall, is a string or bunch of characters surrounded by quotes. For example: NAME$ = "FRED" , D1$ = "FLOWER AND TREES"

The only thing you have to do with the Shell Game strings is to put your quiz problems inside them. Just think of yourself as being so famous that people insist on quoting everything you say.

3. You have 5 empty strings you may fill for each problem. (You may not need all five.) They are called D1$, D2$, D3$, D4$, and D5$.

4. You must use D1$ for EVERY problem. Apple knows it has run out of problems when it finds a D1$ with nothing in it.

5. While you only use as many of the five strings as you need, the strings must be in ascending order; i.e., D4$ may not be followed by D3$. It is not necessary to use each string in order; you may, for example, use D1$, D2$, and D4$ for a given problem without using D3$ at all.

We will talk more about what information you place inside the strings after completing our exploration of the external structure of the lines. Please keep in mind that we are still discussing things that all The Shell Games have in common.

6. There is always a RETURN statement following the last string used. In this case, a :RETURN on line 9011. But it would be equally effective as,

```
9012 RETURN
```

or the RETURN could be on line 9013 or 9014 -- anywhere will do as long as the RETURN appears after the last string.

7. All line numbers are used. So -- after you have typed your problem into the strings, and after you have put in a RETURN statement -- should you have any unused lines remaining, type in each of those line numbers followed by REM and a carriage return, just as was done above on lines 9012 through 9014. It is unnecessary to type any spaces or characters after the word "REM".

8. The final feature of all Shell Game problems is that the final problem is not a problem at all. It is a "dummy" RETURN statement, a line number which would normally be the first line of the next problem with nothing on it but the word "RETURN". It is called the Data Table Terminator and is always a multiple of 5. Example:

```
9085 RETURN
```

This completes the entire external anatomy of all Shell Game problems. We will now discuss the structure of a Match Machine problem and then enter a new problem into the program.

# SPECIFIC TO THE MATCH MACHINE

There are two parts of The Match Machine with which we are concerned. These are the menu which appears following the instructions and the match quizzes themselves.

There are up to four different sets of information with which you will supply your Apple for each quiz you create.

1. The title of the quiz that you wish to appear in the menu. This is mandatory.

2. The headings for the two columns to be matched, tied together with an ampersand (&). This is mandatory.

3. The matches themselves, each match tied to its answer by an ampersand, each group separated by commas. The number of matches is optional.

4. Extraneous answers. These are optional.

If you look at the listing on your Apple or the listing on page 8, you will first see the title:

ANIMALS AND THEIR YOUNG

as it appears on the menu at the beginning of the program. This is followed by the column headings,

ANIMAL&ITS YOUNG

and the individual matches

BEAR&CUB,BIRD&NEST,

etc. Following the matches are some extraneous names of baby animals to make the right column longer than the left. These are,

BUNNY,KIT,ELVER

(All spaces count in The Match Machine! Unless you want to see a space preceding a word on the screen, don't enter it in the problem.)

All in all, you must enter, in order:

1 Quiz title.....34 characters maximum
1 Pair column headings.....15 characters left column word,
18 characters right column word
plus up to:
10 Pairs of matches.....15 characters each left column word,
18 characters each right column word
and you may enter:
4 Extraneous answers.....18 characters maximum each

The following special symbols are used:

Ampersand (&) .....Connects the matches
Comma (,) .....Marks the end of a match set (or a title)

You need have only as many matches as you wish and need not have any extraneous answers at all if you do not wish.

You should think of the 5 strings, D1$ through D5$, as really being one continuous string of characters which happens to be broken into 5 easy pieces. When Apple absorbs a problem, it ties the strings together again into one piece, so it really cares little how you have divided them. The only rule you must observe, upon dividing up your problem among the strings, is that your breaks between strings always occur between sets of matches, where you will normally be placing a comma. This way the individual matches, such as FISH&FRY, FRUIT&APPLE, ONE&UNO, etc., are not split apart.

For example,

```
9011 D2$="ALLIGATOR&PEAR,"
9012 D3$="CROCODILE&TEARS"
```

```
9011 D2$="ALLIGATOR&PEAR"
9012 D3$=",CROCODILE&TEARS"
```

```
9011 D2$="ALLIGATOR&PEAR"
9012 D3$="CROCODILE&TEARS"
```

are fine, while

```
9011 D2$= "ALLIGATOR"
9012 D3$= "&PEAR,CROCODILE&TEARS"
```

will tend to confuse poor Apple and make the first column of matches look very strange indeed.

Apple does not care where or even whether you include the comma at the end of a string. In the examples above, you can see that the comma may be at the end of the first string, the beginning of the second string, or be eliminated completely.

# ENTERING A NEW MATCH PROBLEM

We are now ready to enter a new match quiz. To do so, let us first delete the old by typing:

```
DEL 9010, 9014
```

(Remember that you cannot lose the original version of any Shell Game unless you purposefully SAVE the new version on top of the old. So while we are going to modify the version now inside the computer, the original remains safely undisturbed on your diskette.)

This will clear out the old animals problem and give us five new lines to use. The title of the new quiz as it will appear in the menu is "ELEMENTARY MATH". We will want the screen to appear as shown below when the match quiz is completed:

```
PROBLEM              ANSWER
3 + 4.............7
15 - 6............9
21 / 3............7
2 X 4.............32 / 4
```

So the matches are to be 3 + 4 and 7, 15 - 6 and 9, 21 / 3 and 7, 2 X 4 and 32 / 4. The extraneous answers are to be 9, and 17 + 5.

To enter the problem, type

```
9010 D1$="
```

and then start typing your information, stopping before the end of the third line. For example, type

```
9010 D1$="ELEMENTARY MATH,PROBLEM&ANSWE
R,3 + 4&7,15 - 6&9,21 / 3&7"
```

Note: Apple will automatically start printing on the next line when you run out of space at the end of a line on the screen. Do not press RETURN until you are finished with the entire string.

It would be possible to include more matches in this string, but there are four more empty strings available, so it is not necessary to crowd them.

```
9011 D2$="2 X 4&32 / 4"
```

It is now time to finish up with our extraneous answers. These could easily follow upon the heels of 32 / 4 in D2$, but we may also continue with

```
9012 D3$="9,17 + 5"
```

We then have a RETURN statement:

```
9013 RETURN
```

And finally we use up the last line:

```
9014 REM
```

You may now RUN the program. You will see a new category at the top of the menu,

```
1 ELEMENTARY MATH
```

If you select 1, you will see our new match quiz appear. If all does not appear well, check lines 9010 to 9014 against the original lines above. When you have finished looking, exit the program with CTRL-C . If you want to SAVE this Elementary Math program, now is the time to do so. In this case, however, you would be doing so at the expense of losing the original first match quiz. The next example we will enter, using the Editor, will be a new quiz in addition to the ones currently in the program and will be a better candidate for permanent storage.

NOTE: The Match Machine will correctly link up duplicate matches. If you have, for example, BEAR&CUB and TIGER&CUB, Apple will not care which CUB is matched with which animal.

# ENTERING PROBLEMS

The problem entry scheme just discussed works very well when you have deleted the old series of problems and are typing in a whole new set. But what if you just wish to add one problem onto the end of the existing set? First, you have to find the end of the existing data table. Then you have to make sure you create a new Data Table Terminator line (principle 8). Now, all this is not the end of the world, but why do it yourself when your Apple is perfectly happy to do it for you? Presenting...

# THE PROBLEM ENTRY EDITOR

We are now going to enter a new quiz using the Problem Entry Editor. The subject of this quiz is Animal Classification. If you will wish to permanently add this quiz to The Match Machine, it is suggested that you reLOAD your original copy of the program as you undoubtably do not also wish to keep the little math quiz we have just typed in.

Inside each of The Shell Games is a hidden program designed to make problem entry as simple as possible. Select the Match Machine program from the main menu, and then go to the "hidden" program by pressing

CTRL-C

and then typing

RUN 2

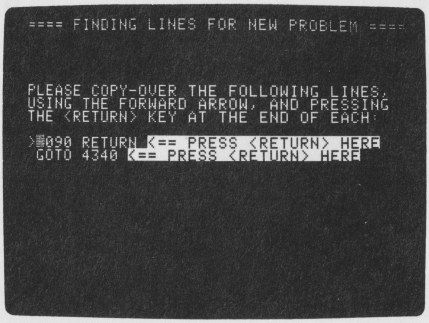
# ENTERING A NEW PROBLEM

You are now in the Problem Entry Editor. Select

1) ENTER A NEW PROBLEM

from the menu. The first thing that occurs is that your Apple will leaf through all the first lines of the existing problems, looking for the Data Table Terminator line: the first line of a problem (an even multiple of 5) containing only a RETURN statement. It knows this because D1$, which was emptied right before it went to the problem, is still empty upon its return. (This is why it is necessary to always use D1$ in each of your quizzes.)

The second thing that usually happens is that the Apple will ask you to copy over a couple of lines, pressing the RETURN key at the end of each, as seen in the photograph on this page:




The cursor will be positioned at the beginning of the first line. Just use the forward arrow key to step across each of the characters, until you reach the space just before the arrow in inverse mode (black text on a white background). Then press the RETURN key. This will position the cursor at the beginning of the next line. Step across the second line using the forward arrow key and again press the RETURN key just before the arrow in inverse mode.

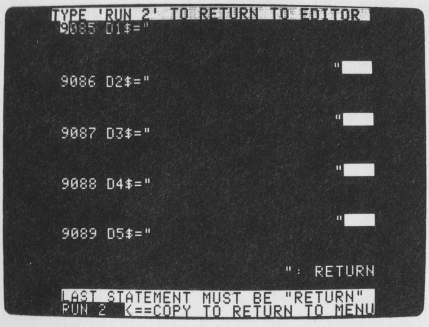
As the new problem will be written at the current end of the data table, on top of the line that now says <linenumber> RETURN , we need to write a new Data Table Terminator line five lines ahead. That is what you are doing.

The reason your Apple doesn't always stop and request you copy lines is that there may already be a RETURN statement 5 lines ahead. People tend to leave Data Table Terminators laying around rather casually. Your Apple doesn't mind -- as long as there is one, there may be one hundred.

After copying over the lines, you will be presented with a screen at the bottom of which you are offered two choices:

<ESC> FOR EDIT, <RETURN> TO MENU

If you had embarked on this course accidentally, you could press the RETURN key to return to the Problem Entry Editor menu. As we came to this spot with malice aforethought, please press the ESC key. You will now see the screen on the next page:




(If the line numbers do not correspond, you simply have a version of the program with more or fewer quizzes already entered. The specific line numbers you see are unimportant.)

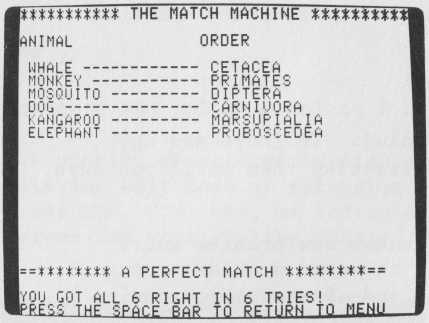
Your Apple has laid out the five strings in their maximum size and placed the cursor at the beginning of the first string. We are given just short of four lines for each string, at the end of which is the close-quote, followed by a white rectangle. The second quotation mark is there to indicate the absolute maximum length the string may be; it is rare that your match will fill-out exactly to just fit. You may enter less than the maximum length by simply closing the string with a quotation mark, followed by the familiar pressing of the RETURN key. In no case may you exceed the limit of the quotation mark. If you do, you will receive a

*** TOO LONG ERR

message and have to re-enter the line. The last string is followed by a RETURN statement. This is the longest the last line containing the RETURN statement may be without causing the error message.

It is important to keep in mind that displaying these maximum string sizes and the RETURN statement on the last line are a way of reminding you of limitations and requirements. In actual operation you would usually need fewer than the maximum number of strings and less than the maximum string length. And you may put the RETURN statement anywhere as long as it follows the last string used.

We may now proceed to enter our new quiz. The title to appear on the menu is "ANIMAL CLASSIFICATION" and we will want the display at the end of the quiz to appear as shown on the next page:




To enter the match, first press the forward arrow key and the REPT key to move the cursor just beyond the first quotation mark on the first line. We may now begin to enter the quiz. First the title, then the column headings, then the matches themselves, and finally the spurious answers.

For example, you might enter:

```
 9085 D1$="ANIMAL CLASSIFICATION,
ANIMAL&ORDER,WHALE&CETACEA,MONKEY
&PRIMATES,MOSQUITO&DIPTERA"
```

(Since we didn't fill the entire allotted space in the string, be sure to include the final quotation mark following the last character in the string. Then use the backspace key to move the cursor to the next numbered line and continue typing.)

```
 9086 D2$="DOG&CARNIVORA,KANGAROO
&MARSUPIALIA,ELEPHANT&PROBOSCEDEA
"
 9087 D3$="EDENTATA,RATITAE"

 9088 RETURN

 9089 REM
```

We could have crowded almost the entire quiz into the first string, but we may wish to come back at a later time and lengthen the title, add new matches (which must occur before the extraneous answers), or make other changes which require increasing the amount of material. It is far easier to do this if we have left some room originally.

The next thing you would normally do would be to follow the screen's advice to RUN 2, but do not do so now as we have not yet covered any of the other features of the Problem Entry Editor.

Please check over the lines you have entered for accuracy and then type

```
RUN
```

to RUN the program and see your work revealed. If there are any problems, you may wish to refrain from correcting them until you have read the next section on editing.

As you can see, the Problem Entry Editor makes new problem entry easier by finding the proper location for the next problem, doing some of the typing for you, and providing end-of-string and don't-forget-the-RETURN-statement reminders. But it takes time for the Problem Entry Editor to find the location of the next new problem. If you are entering an entirely new set of quizzes all at once, after deleting the entire data table by typing:

```
DEL 9010,9999
```

you may find it a lot faster not to use the Problem Entry Editor. If you are doing all new problems, you know on which line the next problem will occur, and it can often be faster to enter it "by hand", perhaps using the AUTO command to supply you with line numbers. (See the Apple BASIC Programming Manual, pages 65-67.) Try it both ways and see which you prefer.

# EDITING AN EXISTING PROBLEM

If you have been following along on the computer, you will now have a match-pair consisting of ELEPHANT and PROBOSCEDEA. Unfortunately, PROBOSCEDEA is mispelled. The correct spelling is PROBOSCIDEA. (To the 3 people who knew that as soon as they saw the word, I am envious.) There are several ways to correct this problem, among them being a complete retyping of line 9086, using ESC D sequences to get up to line 9086 and then copying over the string, using ESC A 's to skip over the spaces. Another way would be to type

```
POKE 33,33
```

before LISTing the line to keep the extra spaces from occuring, then typing

```
TEXT
```

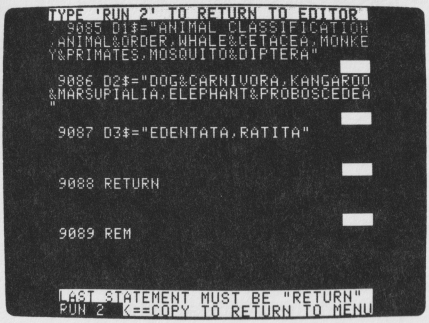
to restore the normal mode after copying the line. All much too much work.

Type

RUN 2

to enter the Editor. Select 2 from the menu and press the RETURN key.

The problem we entered is displayed and we are given the option of entering edit mode or returning to the menu. Since we wish to edit, press the ESC key, as indicated at the bottom of the screen. The screen now appears like this:




The lines have been moved over to the left so that no extra spaces exist on the left, relieving us from having to type any ESC A 's to leapfrog over them. The cursor has been placed at the beginning of the first string. You may enter as much text as will fit between the beginning of each string and the small white rectangle; in fact, the cursor can appear on top of the first block of the small rectangle when you press RETURN . In the match quiz we are entering, we fall safely short of these maximums.

As we wish to operate on the second line, press the backspace key four times. This will move the cursor down one row at a time until it is on top of the 9 . Remember this technique: whenever you wish to go down the screen one line at a time and the cursor is on the left side of the screen, use the backspace key.

After arriving at the first line of D2$, press the forward arrow, followed by the REPT key, until the cursor is resting over the first E in PROBOSCEDEA. Press I to change the E to an I and then press the forward arrow four times so that it is on top of the first space following the quotation mark. Then press the RETURN key. The new line has now replaced the original.

During the time that you are in Edit mode or Enter New Problem mode, you are not being controlled by The Shell Game. The Problem Entry Editor has set up the window and printed helpful information on the screen, but you are esentially talking directly to Apple. At this point, it is important for you to be able to use the Apple editing features: the forward arrow and backspace keys and the various ESC sequences -- ESC A , ESC B , etc. With them, you may do extensive editing of problems with speed and ease. Take the time to practice until you are proficient.

If at any time during the entry of data or the editing of data, you find the screen becoming cluttered or you get an error message that is covering up part of what you are doing, type

RUN 2

again and select

2) EDIT CURRENT PROBLEM: #<problem number>

This will clear the screen and redisplay the lines you have already sucessfully entered.

You may now either type

RUN 2

or backspace to the bottom line and copy over RUN 2 with the forward arrow. Either way will return you to Problem Entry Editor control and the menu.

# DISPLAYING A PROBLEM

We may now check to see that the problem is corrected and will be properly displayed. To do so, select

3) DISPLAY CURRENT PROBLEM: #<current problem number>

from the menu. This selection will display the quiz in the normal way, with the exception that we have the option of instantly returning to the Problem Entry Editor by simply pressing the ESC key. You may now play the game to confirm that all is well or return to the Problem Entry Editor immediately.

If you wish to confirm that all is well with your title, RUN the program in the normal way and check that it appears correctly in the menu.

We will not be referring to this newly entered problem again. If you are tired of reading and want to come back later, you need not keep the program in the computer any longer. You may LOAD the original version again later. If you wish to keep this version of the program containing the new quiz, you may now SAVE it on your diskette the same way you SAVE any other program you have written.

Insert the diskette upon which you want the quiz to appear, and type:

SAVE <program name>

Then press RETURN . <Program name> may continue to be The Match Machine, or you may change the title to anything you wish.

# SEARCHING FOR A PROBLEM

One of the prime reasons for the creation of the Problem Entry Editor was the difficulty encountered when trying to find a particular problem somewhere in the middle of the data table. Listing forty or fifty problems, in the case of Professor True, for example, in order to find the problem with a typographical error in one word can be most distressing. There are two features of the Problem Entry Editor that were designed to alleviate this problem.

First, in all The Shell Games, the Problem Entry Editor is aware of where you are in the program whether you have been using it or not. Exit the Problem Entry Editor (select 7 END) and type

RUN

Select category 10 from the menu and let the matches be displayed. You might be playing this game and suddenly discover that QUAIL is spelled QVAIL. Press CTRL-C and type

RUN 2

The Problem Entry Editor indeed knows you are working on category 10 and is prepared to instantly set up the program lines for editing. (It's all done with mirrors!) This is true of all The Shell Games.

Of course, there will be occasions when you do not wish to edit the particular problem you have been displaying. So the Problem Entry Editor has a search mode to allow you to find the problem on which you do wish to work.

Select

4) SEARCH FOR A PROBLEM TO BE EDITED

You may begin your search from one of three places: the beginning, the current category, or the end of the table. When looking for a particular problem, make your best guess as to which of these three locations is closest to the problem for which you are searching and select that option. As

3) THE END OF THE DATA TABLE

is the most fun, please select that now.

The Apple just went forward from the last Current Problem to the the Data Table Terminator. The line at which you are looking is the Terminator: a line number that is an even multiple of 5 with only a RETURN statement after it. Pressing the left-pointing arrow key will move you back to the penultimate problem. (Excuse the outbreak of sesquipedalianism, but as the life science teachers in the audience were allowed to be in on the PROBOSIDEA trick, it seems only fair to give the English department a chance to strike back.)

By using the left-pointing arrow key to move back through the problems and the right-pointing arrow key to move forward, coupled with the REPT key for speed, you may quickly find any problem in memory. Note, however, that only the first line of each problem is displayed. Experimentation proved this to be the most efficient scheme in terms of speed and clarity. Once you have found the problem you wish to edit or display, press the RETURN key. This will take you back to the main Problem Entry Editor menu. The Problem Entry Editor will now consider the problem you were viewing at the time you pressed RETURN to be the new Current Problem and be prepared to let you edit or view it.

# VERIFYING THE DATA TABLE

As simple as the structure of the problems may now seem, it is quite easy when entering a lot of problems to make mistakes, such as not terminating each problem with a RETURN statment or forgetting the Data Table Terminator. Some of the errors created by these minor slips are very difficult to understand. Not having a RETURN statement at the end of a problem, for example, will cause the next problem to occur twice! As it is Apple's job to be concerned with such matters, verifying has been automated and your Apple will gladly do it for you if you will select

5) VERIFY DATA TABLE FORMAT

If there are any problems, your Apple will announce exactly what they are and help you to correct them, usually printing out the corrected lines and asking you to copy them with the forward arrow key, as we did before during the exercise on Entering a New Problem. After you have made the correction, the Problem Entry Editor will verify that the correction was done properly, asking that you do it again if it is wrong. (Do not be frightened by this; it is very difficult to do it wrong. The check is there mainly to give you confidence that you definitely did it right.)

The only type of error to which the computer cannot dictate a correction is a missing RETURN statement. If you have forgotten the RETURN statement at the end of a problem, the Problem Entry Editor will tell you so and bring the program to the problem that lacks the RETURN statement, leaving you in edit mode. You may enter a RETURN statement following the last string used and thus resolve the problem. Afterwards, return to the Problem Entry Editor and verify that the data table is now correct.

When the problems are cleared up, your Apple will give the data table a clean bill of health and announce that you should press the RETURN key to return to the menu.

We shall not dwell on this selection any longer; the data structure of the problems is now familiar to you. There are no obscure errors; there are the eight principles to the problem anatomy and no more. If these are met, your Apple will be happy and the program will run.

# REPLACING AN OLD PROBLEM

To replace one problem with another, search for the problem, select the editing option, and delete the lines of the problem. Then re-enter the Problem Entry Editor and select

1) ENTER A NEW PROBLEM

The empty strings will be displayed and you may proceed to enter your new problem. If this is your first time through the Problem Entry Editor procedure, remember that you must actually enter a new problem! You may not leave a gap in the data table! If you do, you are breaking two of the cardinal rules of the Shell Games: all Shell Game problems occupy exactly five lines, and all line numbers must be used.

# INSERTING & DELETING PROBLEMS

You can both insert and delete problems from Shell Game data tables. However, these are not tasks for a brand new programmer. Until you are up to speed, it is suggested you skip this section. Full use of the Games may be made without doing insertions and deletions.

Both inserting and deleting depend on the Integer Basic Renumber program. This program resides in the Programmer's Aid #1 ROM and can be found in a softer form in the Contributed Software Bank Vol 5, both available from your Apple dealer. The instructions for using the Renumber program are provided in the accompanying documentation.

For each new problem you wish to insert, renumber the data table starting from the point at which you wish to insert a problem to 9990 with an increment of 1, freeing 5 lines for your insertion. To delete a problem, delete its lines and then renumber starting from the problem following the one you wish to delete to 9990 with an increment of 1, using up the 5 lines that are left free. For example, to insert a problem before problem 5, which currently resides from 9025 to 9029, you would enter:

```
CLR
START = 9035
STEP = 1
FROM = 9030
TO = 9990
CALL <the renumber-part-of-a-program address>
```

Conversely, to delete problem 5, enter:

```
DEL 9030, 9034
CLR
START = 9030
STEP = 1
FROM = 9035
TO = 9990
CALL <the renumber-part-of-a-program address>
```

The specific CALL address you will use depends on which version of renumber (RAM or ROM) you are using and, in the case of the RAM version, where you have located the program. Consult the manual supplied with the Renumber program for the address of the CALL.

This insertion and deletion scheme depends upon each and every line in the data table being used to prevent compression. If only 4 lines were used for a problem, the next problem's first line would suddenly become the last line of the previous problem, creating chaos and ruin. This is the reason for the principle that each line must be used and for the menu selection,

6) VERIFY TABLE FORMAT TO ALLOW RENUMBER

which, in addition to checking everything covered by selection 5, also confirms that all lines are used. If any unused lines are discovered, the program will print the lines necessary to correct the problem and ask that you copy them.

WARNING: It is vital that you do this verification before attempting to insert or delete, as "decompression" is frustrating and time-consuming.

Insertion and deletion could have been fully handled by the Problem Entry Editor. But to do so whould have required considerably more program space. As they are functions that are seldom necessary, and many of the programs are already butting up against 4 8K, it seemed a good compromise to have the Problem Entry Editor handle the vital line checking and the user handle the renumbering itself.

# LEAVING THE EDITOR

Selection

7) END

is the preferred way to exit the Problem Entry Editor, as it will remind you to SAVE the program if you have made any changes.

WARNING: Not SAVEing your program will result in the loss of all the changes and additional quizzes you have entered.

The normal procedures for saving programs work in the same manner here as anywhere else. They have been described in the previous section called "Displaying a Problem".

# AN IMPORTANT FRINGE BENEFIT

When a Shell Game is ended, it is deleted and the Index program is run. This is most distressing if you have changed one or more problems but have not yet SAVEd the new version, as it will suddenly cease to exist. When you type

```
RUN 2
```

the first thing that occurs is that this return-to-index routine is disabled so you may SAVE at your leisure. When you are completely through with the program, you may type

```
RUN INDEX
```

to get back under disk control once more.

WARNING: There are also other more subtle protective measures taken to protect your work from harm, so it is a good idea to always enter the Problem Entry Editor momentarily before working on the program, even if you will be entering problems "by hand".

# FITTING YOUR PROGRAM NEEDS

Most of The Shell Games have options you may exercise to change the way the program runs. To see these option "flags",

```
LIST 9000, 9009
```

In the case of The Match Machine there is only one option; Professor True and Mr. Multiple have a number of options which allow fundamental changes to be made as simply as the turning of a swich.

Line 9005 of The Match Machine reads:

```
9005 LESSFLASHINGFLAG=NO
```

or it reads:

```
9005 LESSFLASHINGFLAG=YES
```

Each time the user answers a question, there is a "reward" given by Apple in the form of flashing and sounds. The amount of color and sound displayed is lowered appreciably by saying YES to LESSFLASHINGFLAG. Try it both ways.

This brings us to the end of the discussion of The Match Machine. In the next section, we will discuss the formatting of Professor True and Mr. Multiple, and how they differ from The Match Machine. If you have been reading and experimenting straight through, it is suggested you take a break before proceeding.

# INSIDE PROFESSOR TRUE

It is important that you go through the discussion on The Match Machine before reading this section, as most of the things you will need to know to use Professor True are explained in the section on The Match Machine. Likewise, most of the further details you will need to enter a quiz in Mr. Multiple will be explained here. So please do read this manual in order.

Everything but what you put inside the strings is the same in all Shell Games. So you already know almost everything you need to know to use Professor True. The problems look the same, and you will find the same Problem Entry Editor by typing

RUN 2

Before examining what you put into the strings, let us examine the overall program.

The Following Is Dull and Boring and To Be Read By Educators Only:
(Jump down to "DATA STRUCTURES & SCARY STUFF".)

Professor True is a highly flexible true/false quiz shell. As with Mr. Multiple, its reason for being is to teach, not to test. There is no way to use this quiz structure for graded examinations. You may, however, require that all questions be answered correctly before the user can exit the program, thereby creating a quiz on which all students will get 100%. (The way to do this will be discussed in one of the following sections called "MODIFYING PROFESSOR TRUE".)

There are two differences between Professor True and Mr. Multiple: First, there is the rather obvious difference of Professor True being a true/false quiz and Mr. Multiple being a multiple choice quiz. The other difference is that Professor True was designed to be used by much younger people. The number of words you may use within a single problem is far fewer, the number of choices the child can make is by definition restricted to two, and there is an intensive amount of stroking and positive reinforcement. None of this, of course, restricts its use by older people.

In both programs, the color, sound, and the game format, with little pieces of information being digested easily, keep the player's interest and attention.

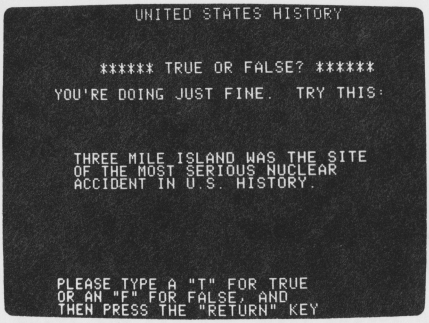
# DATA STRUCTURES & SCARY STUFF

In the Match Machine, the basic unit of information was a match-pair. In Professor True and Mr. Multiple, the basic unit is a word. (No, not a computer word, a real-live English word, like "the" or "hippopotamus".) There are some new special symbols to be learned, but you will find things pretty much the same as in The Match Machine.

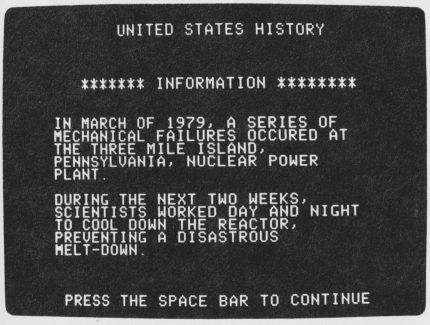
The special symbols for Professor True are:

Asterisk (*)..... the following question is true
"At" sign (@).....the explanation begins
"At" sign (@).....a new paragraph in the explanation begins

Let us enter a new problem in Professor True that will look like this:




The Question




The Answer

Having LOADed Professor True, enter the Problem Entry Editor with CTRL-C and then type

```
RUN 2
```

and select

1) ENTER A NEW PROBLEM

We may now begin typing in the question, starting (as the answer is true) with an asterisk (*). The asterisk must be the very first character of the problem. After this we may begin typing the question, ending the first string when it is convenient. You may end anywhere as long as it is not in the middle of a word. Anywhere. For example, as far as your Apple is concerned,

```
9210 D1$="NOW IS THE TIME"
9211 D2$="FOR ALL GOOD MEN TO"
9212 D3$="COME TO THE AID OF THEIR"
9213 D4$="PARTY."
```

is the same as,

```
9210 D1$="NOW IS THE TIME FOR ALL GO
OD MEN TO COME"
9211 D2$="TO THE AID OF THEIR PARTY."
```

Your Apple has been trained to add a space where each pair of strings join so that you do not have to worry about having a space (referring to the last example above) after "COME" or before "TO".

After you have typed the complete question, type an "at" sign (@) and begin typing the first paragraph of the answer, again stopping when you begin to near the second quotation mark. When it is time for the second paragraph of the answer, type another "at" sign and continue typing. When you are through, be sure to end with a return statement. If you have typed it in correctly, it will look something like this:

```
 9210 D1$="*THREE MILE ISLAND WAS
 THE SITE OF THE MOST SERIOUS NUC
LEAR ACCIDENT"

 9211 D2$="IN U.S. HISTORY.@IN MA
RCH OF 1979, A SERIES OF MECHANIC
AL FAILURES OCCURRED"

 9212 D3$="AT THE THREE MILE ISLA
ND, PENNSYLVANIA, NUCLEAR POWER P
LANT.@DURING THE NEXT TWO WEEKS,
SCIENTISTS WORKED DAY"

 9213 D4$="AND NIGHT TO COOL DOWN
 THE REACTOR, PREVENTING A DISAST
ROUS MELT-DOWN."

 9214 RETURN
```

(The line numbers in your program may vary from those above.)

The breaks between strings need not duplicate the ones in the example above, as long as those breaks do not occur in the middle of a word. Please note that the paragraph symbols do not have any spaces around them. If you wanted to indent the paragraphs, you could put 3 to 5 spaces after each @.

It is strictly voluntary to have explanations at all; if you do not wish an explanation, just stop typing at the end of the question.

Enter the Problem Entry Editor again and display your problem (select

3) DISPLAY CURRENT PROBLEM

from the menu.) If there is any problem, compare your entry with the example above.

* Do you have an asterisk as the very first character? If not, the Apple will think the answer is false.

* Do you have spaces surrounding an @? If so, you will see spaces on the screen.

* Have you broken a word in the middle? If so, your Apple will put a space in it.

* Did you put a RETURN statement on the end? If not, the Apple may display the wrong problem or no problem at all.

* If something serious happens, such as the program ceasing to operate entirely, run the Problem Entry Editor and select

5) VERIFY DATA TABLE FORMAT

from the menu to verify that the basic anatomy is all right.

And that is all there is to it! You may wish to exit the Problem Entry Editor with selection 7 and then SAVE this new version of the program. You may wish to enter even more current events and save an even more up-to-date version.

When you wish to create a completely new true/false series, type:

DEL 9010, 9999

to clear out the current quiz. Then either start entering new questions "by hand" beginning on line 9010, or use the ENTER A NEW PROBLEM feature in the Problem Entry Editor. Keep your questions, including the answer, down to around 25 to 30 words and type away. You may then proceed to RUN the game and enter the Problem Entry Editor if you discover any problems. The original American History quiz of 41 questions took around 20 minutes to enter and another 15 to edit. How long it took to write is another matter entirely.

# MODIFYING PROFESSOR TRUE

Professor True (and Mr. Multiple) may be extensively modified to fit your needs and educational philosophies. No programming is required. Please press CTRL-C and then type

LIST 9000, 9009

to see the program controls. First, you are given 5 strings, D1$ through D5$, in which you may enter lines for your quiz's title page.

Unlike the problem strings, each of these strings is a separate entity. Each will hold one line (38 characters) of information, and each line will be automatically centered on the title page screen. You need use only as many lines as you wish.

Following these strings, you will find the only feature that is not found in Mr. Multiple: PROFSPEAKS. PROFSPEAKS may be set to any number from 0 to 10, and sets the frequency with which Professor True will give the user encouragement. 0 is never, 10 is on each and every problem. You may select the number based on the age group you are trying to reach, your own philosophies, or based on the amount of encouragement you have written into the questions themselves. As the Professor, while speaking, tends to give out extra points rather freely, the higher the setting, the more points the user may get in playing the game.

NOTE: In the following FLAGS, EXPLANACION and SCORING are mispelled to avoid having your Apple choke on the reserved words, AT and OR, contained in the properly spelled versions.

SKIPEXPLANACIONIFRIGHTANSWERFLAG=YES will cause the program to give the further information only if the user missed the question. This can be very useful when the material in the quiz is not new to the student and the explanation is essentially a review. You may wish to allow the students who know the subject matter to breeze through the quiz, leaving more time for students whose knowledge is a little shakier. This flag has no effect on the second round of the quiz, when the user is retaking the questions missed: the program will always skip the explanation if the question is answered correctly.

Please keep in mind that these flags can be changed at any time: it may be that the first time you have people take the quiz the information will be new and you will wish them to read all of it. A few weeks or months later, you may want the same people to take the same quiz, but this time have the explanation skipped if they are right, or use some other variation below. Just change the answers to the flags from NO to YES, or YES to NO and you have an entirely new format for the same quiz.

ALLOWEXPLANACIONSKIPFLAG=YES will allow the user to prematurely press the space bar and skip the explanation before it has been completely printed.

SKIPEXPLANACIONFLAG=YES will force the skipping of any explanations. There is no option given to the user; as far as the user is concerned, none of the quiz problems have any explanation section.

SKIPSCORINGFLAG=YES will eliminate scoring and make PROFSPEAKS = 0

MUSTANSWERALLQUESTIONSFLAG=YES will eliminate the option of using the ESC key to skip the balance of the questions. Every question will have to be answered before the user is let out of the program. They need not all be answered correctly, however.

MUSTRETESTWRONGANSWERSFLAG=YES will cause the user to have to go back and take over all missed questions before exiting the quiz. It does not force the user to take all the questions in the first place.

The combination of MUSTANSWERALLQUESTIONSFLAG=YES and MUSTRETESTWRONG-ANSWERSFLAG=YES will create a quiz in which each and every question must be answered correctly before the game ends.

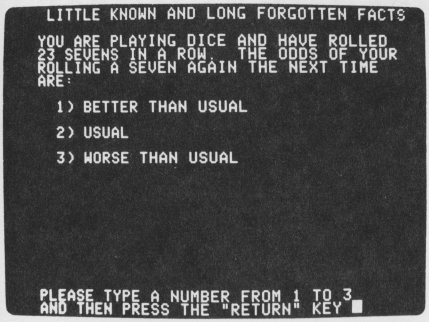
By setting these flags to YES or NO, you can customize the quiz to your own purpose. For example, if you wish to be sure that your students have indeed learned a quiz they have been allowed to take at their leisure, you might set the flags to require their taking every question and getting each one right. You might or might not set the SKIPEXPLANACIONSFLAG; you might or might not allow the students to voluntarily skip the explanations with the ALLOWEXPLANACIONSKIPFLAG set to YES.

When you set these flags, everything in the program will be made to correspond, including the instruction pages. This is the reason there are no real instructions given in the manual as to how to use these quizzes; you will be deciding what those instructions will be.

# INSIDE MR. MULTIPLE

If you have been following along from program to program, you are about to find that the most sophisticated program in The Shell Game series is the easiest to learn.

Let us enter the following multiple-choice question and answer:




The Question


The Answer

The special symbols for Mr. Multiple are:

```
Semicolon (;).......this is a possible answer; it will be
                    numbered and indented.
Asterisk (*)........this is the first character of the
                    correct answer.
"At" sign (@).......the begining of the explanation
                    (same as TRUE).
"At" sign (@).......the begining of a new paragraph in the
                    explanation (same as TRUE).
Semi-colon (;)......the explanation will end with another
                    list; also numbered and indented.
```

As you can see, the symbols, with the exception of the semicolon, are the same as for Professor True. The asterisk has a slightly different meaning now: in Professor True it meant the whole question was true; here, it is pointing out the right answer among all the possible answers.

After LOADing Mr. Multiple, and getting to the Problem Entry Editor; select

1) ENTER A NEW PROBLEM

and enter our new problem, keeping in mind that we have much more information to type in than in any of the other programs so we no longer have "string space" to burn. Fill each string to its capacity and do not forget the :RETURN statement at the end of the problem.

When you are done, your problem will look something like this:

```
 9135 D1$="YOU ARE PLAYING DICE A
ND HAVE ROLLED 23 SEVENS IN A ROW
.  THE ODDS OF YOUR ROLLING A SEV
EN AGAIN THE NEXT TIME ARE:;"
 9136 D2$="*BETTER THAN USUAL;USU
AL;WORSE THAN USUAL@GAMBLERS WILL
 INSIST THAT HITTING A SEVEN AGAI
N IS ALL BUT IMPOSSIBLE.@"
 9137 D3$="MATHEMATICIANS WILL EX
PLAIN THAT EACH ROLL IS A SEPARAT
E ENTITY AND THE ODDS ARE NO DIFF
ERENT FOR THE 24TH ROLL THAN"
 9138 D4$="THEY WERE FOR THE FIRS
T.@PRAGMATISTS, HOWEVER, WILL COR
RECTLY POINT OUT THAT THE ODDS AR
E YOU WILL CONTINUE"
 9139 D5$="THROWING SEVENS BECAUS
E THE ODDS ARE THERE IS SOMETHING
 SERIOUSLY WRONG WITH THE DICE.":
 RETURN
```

Enter the Problem Entry Editor and display the problem, clearing up any errors using the edit mode. Remember that if the screen gets messy, by RUNning 2 and selecting 2 (edit mode), you get a fresh start.

As with the other Shell Games, if you wish to retain this question in the quiz, exit using selection 7 and SAVE the new version.

# GENERAL COMMENTS

With questions approaching 100 words in length, it can get pretty crowded inside 5 strings. So keep your questions and answers as short as possible. You will have an easier time entering them and an easier time keeping your audience's attention.

Considering how few special symbols there are associated with Mr. Multiple, there are an amazing number of formats you may use for your questions. It is worth exploring some of them: go to the search mode and select problem #1 as the current problem. Then go through the program examining the questions. When you don't understand how one was done, ESC back to the Problem Entry Editor and examine the questions in edit mode. (Don't press the ESC key when you get there and you'll be able to pop right back using the RETURN key.)

# MODIFYING MR. MULTIPLE

By typing

LIST 9000, 9009

you will see that Mr. Multiple has the same title page strings and the same flags as Professor True, with the exception that there is no PROFSPEAKS in Mr. Multiple. Please refer to the section called "MODIFYING PROFESSOR TRUE" for an explanation of this area.

# IN CONCLUSION

It is hoped you will enjoy using The Shell Games. It is also hoped you will develop some interesting quizzes. We would like to assemble a library of generally useful games that we can make available to schools and colleges around the country. We would also like to publish some of the more universal quizzes. If you do write something that is of use to others, please send us a copy of it. The address is:

The Software Bank
Apple Computer Inc
10260 Bandley Drive
Cupertino,
California, 95014

We hope to hear from you.

# APPENDIX A:
# FORMATS AND SPECIAL SYMBOLS

## FOR ALL SHELL GAMES

External anatomy: The Eight Principles:

1) The first problem is on line 9010.
2) All problems take 5 lines and follow each other sequentially.
3) All problems have 5 available strings: D1$, D2$, D3$, D4$, and D5$.
4) All problems must use D1$ first.
5) The strings used must be in ascending order, e.g., D1$, D3$, D5$, not D1$, D5$, then D3$
6) All problems have a RETURN statement following the last string used.
7) All problems use all five assigned line numbers, filling in extras with REMs.
8) The last "problem" is the Data Table Terminator, a RETURN statement where the first line of the next problem would be.

Program modifiers:

Lines 9000 to 9009 contain flags that allow you to modify the programs. In addition, there may be 5 strings for changing the title page.

The editor may be reached by pressing CTRL-C and then typing

```
RUN 2
```

To prepare a Shell Game for an entirely new set of problems, first type

```
DEL 9010,9999
```

to clear out the old problems.

# THE MATCH MACHINE

String Data Structure
For each question, you must have:

```
1 Quiz title.............34 characters maximum
1 Pair column headings....15 characters for left column word
                          18 characters for right column word
up to:
10 Pairs of matches.......15 characters for left column word
                          18 characters for right column word
and you may have up to:
4 Spurious answers........18 characters for right column word
```

Special characters:

```
Ampersand (&).............connects the matching pairs
Comma (,).................separates the matching pairs
                          from each other and from title
```

EXAMPLE

To Get This

menu listing:

PHRASES AND THEIR MEANINGS

quiz result:

```
PHRASE                    MEANING
KNOCK IT OFF-----------DESIST
LIKE A SHOT------------QUICKLY
BLUE SKY THE IDEA------THINK ABOUT IT
UP IN ARMS-------------INDIGNANT
BY DESIGN--------------PURPOSELY
SPLIT HAIRS------------QUIBBLE
```

Enter this

```
 9010 D1$="PHRASES AND THEIR MEANI
NGS,PHRASE&MEANING,KNOCK IT OFF&D
ESIST,LIKE A SHOT&QUICKLY"

 9011 D2$="BLUE SKY IT&THINK ABOU
T IT,UP IN ARMS&INDIGNANT,BY DESI
GN&PURPOSELY,SPLIT HAIRS&QUIBBLE,CH
IDE,WARM"

 9012 RETURN


 9013 REM


 9014 REM
```

# PROFESSOR TRUE

String Data Structure
For each question, you must have:

```
1 Question.............maximum total words: around 25
1 Answer...............maximum total words: around 25

Special characters:

Asterisk (*)............as first character, indicates
                        the answer is true
"At" sign (@)...........first occurance means begining
                        of explanation
"At" sign (@)...........subsequent occurances indicates
                        paragraphing in explanation
```

EXAMPLE

To Get This

```
question:

THE LONG COUNT WAS BORN IN 1873
AND LIVED TO THE RIPE OLD AGE OF
105, THEREBY EARNING HIS NAME.

answer:

THE LONG COUNT OCCURRED IN THE
DEMPSEY-TUNNEY HEAVYWEIGHT
CHAMPIONSHIP FIGHT, SEPTEMBER
22, 1927, WHEN THE REFEREE
ALLOWED TUNNEY TO REMAIN ON THE
CANVAS FOR 13 SECONDS WITHOUT
CALLING HIM OUT.

TUNNEY WENT ON TO WIN THE
CHAMPIONSHIP.
```

Enter This

```
 9210 D1$="THE LONG COUNT WAS BOR
N IN 1873 AND LIVED TO THE RIPE O
L DAGE OF 105, THEREBY EARNING HI
S NAME.@THE LONG COUNT"
 9211 D2$="OCCURRED IN THE DEMPSY
-TUNNEY HEAVYWEIGHT CHAMPIONSHIP
FIGHT, SEPTEMBER 22, 1927, WHEN T
HE REFEREE"
 9212 D3$="ALLOWED TUNNEY TO REMA
IN ON THE CANVAS FOR 13 SECONDS W
ITHOUT CALLING HIM OUT.@TUNNEY WE
NT ON TO WIN"
 9213 D4$="THE CHAMPIONSHIP.": RET
URN


 9214 REM
```

# MR. MULTIPLE

String Data Structure
For each question, you must have:

1 Question.....
1 Answer.....Maximum combined total words:
around 90

Special characters:

Semicolon (;).....indicates possible choices, to be
numbered and indented
Asterisk (*).....as first character of a choice,
indicates correct choice
"At" sign (@).....first occurrence means begin
explanation
"At" sign (@).....subsequent occurrences indicate
paragraphing in explanation
Semicolon (;).....indicates list items, to be numbered
and indented

EXAMPLE

To Get This

question:

WHAT IS THE WORLD'S MOST POPULAR
FOOD?

1) BEEF

2) EGGS

3) RICE

4) MILK AND CREAM

5) POTATOES

answer:

MILK IS THE WORLD'S FAVORITE BY FAR.
OVER 145 QUARTS PER PERSON PER YEAR
ARE CONSUMED.

```
HERE ARE SOME OTHER WORLD FAVORITES:

1) 120.2 LBS OF POTATOES

2) 88.9 LBS BEEF

3) 87.7 LBS REFINED SUGAR

4) 263 EGGS WEIGHING A TOTAL OF
   35.3 LBS

5) 7.7 LBS RICE
```

Enter This

```
 9210 D1$="WHAT IS THE WORLD'S MO
ST POPULAR FOOD?;BEEF;EGGS;RICE;*
MILK AND CREAM;POTATOES@"

 9211 D2$="MILK IS THE WORLD'S FA
VORITE BY FAR.  OVER 145 QUARTS P
ER PERSON PER YEAR ARE CONSUMED.@
HERE ARE SOME"
 9212 D3$=" OTHER WORLD FAVORITES:
;120.2 LBS OF POTATOES;88.9 LBS O
F BEEF;87.7 LBS OF"

 9213 D4$="REFINED SUGAR;263 EGGS
WEIGHING A TOTAL OF 35.3 LBS;7.
7 LBS RICE"

 9214 RETURN
```

# APPENDIX B:
# BIBLIOGRAPHY

Every attempt was made to ensure accuracy in the original questions used in The Shell Games. Whenever possible, at least three sources were used to confirm each answer. Where leading sources disagreed, the question was dropped. For example, there is raging controversy over the name of the fastest land animal -- some references claim the Cheetah, some the Pronghorn Antelope. So the question was broadened to include all animals, where there is no controversy -- the Swift is the swiftest. (There were two possible other contenders: The Peregrine Falcon, which has been reliably measured at only 82 MPH, and a strange bug called the Botfly, said to hit speeds of up to 800 MPH, a high enough velocity to penetrate a man like a bullet. Fortunately, there is no credence to this peculiar belief; no known insect can exceed 36 MPH.)

The following is a partial bibliography of the references used in this work:


The Book of Lists -- By David Wallechinsky, et. al., Bantam Books
Citizenship -- D. L. Hennessey, P. O. Box 281, Berkeley, Ca.
The Dictionary of Misinformation -- By Tom Burnam, Ballantine
The Dragons of Eden -- Carl Sagan, Random House
The Earth Shook, The Sky Burned -- Bronson, Doubleday & Co.
The Encyclopaedia Britannica -- Encyclopaedia Britannica, Inc.
The Fabulous Fifties -- Bart Andrews, New American Library
Funk and Wagnalls Encyclopedia -- Funk and Wagnalls, Inc.
Guiness Book of World Records -- McWirter, Bantam Books, Inc.
How Sweet It Was -- Shulman & Youman, Shorecrest, Inc.
History of Radio -- Irving Settel, Grossett & Dunlap, Inc.
History of Televison -- Settel & Lass, Grossett & Dunlap, Inc.
Information Please Almanac -- Information Please Publishing
The Oxford English Dictionary -- Oxford University Press
The People's Almanac -- By D. Wallechinsky, et. al., Doubleday
Prodigal Genius, The Life of Nikola Tesla -- O'Neill, Tartan Books
Roots -- Alex Haley, Doubleday & Co.
The Official Superman Quiz Book -- By Bruce Nash, Warner Books
Scientific American -- various issues
TV Book -- Judy Fireman, Workman Publishing Company
The Trivia Encyclopedia -- By Fred L. Worth, Brooke House
The World Almanac -- Newspaper Enterprise Association
The World Book Encyclopedia -- Field Enterprises

Finally, I feel compelled to explain the strange circumstances under which Lamont Cranston appears twice in the quiz, Secret Identities, in The Match Machine. Lamont Cranston is a world explorer who has given Kent Allard permission to use his name as a secret identity. The Shadow, in turn, is the secret identity of Lamont Cranston, who is really Kent Allard. I hope that clears that up.

Please let us know at The Software Bank how you like the programs we have been publishing and the type of programs you would like to see us publish in the future.


10260 Bandley Drive
Cupertino, California 95014
408 996-1010